



本科生实验报告

实验课程：____操作系统原理实验____

实验名称：____lab9：实现 malloc 和 free____

专业名称：____计算机科学与技术____

学生姓名：____张玉瑶____

学生学号：____23336316____

实验地点：____实验楼 B203____

实验成绩：____

报告时间：____2025 年 7 月 12 日____

Section 1 实验概述

我们已经实现了以页为粒度的动态内存分配和释放。但是，我们在程序中使用的往往是以字节为粒度的动态内存管理机制，即我们可以分配和释放任意字节长度的内存。在本项目中，同学们需要实现系统调用 `malloc` 和 `free`。`malloc` 用于分配任意字节的内存，`free` 用于释放任意字节的内存。在实现了 `malloc` 和 `free` 后，需要自行提供测例来测试 `malloc` 和 `free`。根据测试方法和输出结果来解释自己程序的正确性。最后将结果截图并详细分析 `malloc` 和 `free` 的实现思路。

本实验是在 lab8 的 src6 下做的。

Section 2 实验步骤与实验结果

- 任务要求：实现 `malloc` 和 `free` 并测试。
- 思路分析：在 C 语言里，堆（Heap）是程序运行时可动态分配的内存区域，与之相关的核心函数是 `malloc` 和 `free`。`malloc` 的作用是在堆上分配指定大小的连续内存空间，其返回值是一个指向该内存区域起始位置的指针。`free` 的功能是释放之前通过 `malloc`、`calloc` 或者 `realloc` 分配的堆内存，让这部分内存可以被系统再次使用。

而在我们的操作系统设计中，`malloc` 的返回类型直接设置为 `int` 并返回分配的首地址即可，`free` 得到需要释放的 `byte` 数目后对堆进行释放。

在 `memory` 文件声明并实现新的结构体 `Heap` 和类 `HeapManager`，用于实现堆和 `malloc & free`。

- 实验步骤：

1. 设计结构体 `Heap`。

`Heap` 是我们在 `user pool` 中开辟的一块空间，大小为 10 页，从 `userpool` 的 `start address` 开始分配。这 10 页内存专门用来分配和释放以字节为单位的内存。遵循先进后出的原则，保证分配的内存永远是从首地址开始的连续的一段字节，未分配的内存也永远是连续的。

在 `memory.h` 中声明。

```
struct Heap{
```

```

int count;//分配页数
int startAddr;//分配起始地址
int shengyu;//记录剩余字节数

int tail;//记录 heap 的最后一个地址

Heap();
void initialize();

};

```

在 memory.cpp 中实现。Heap 大小固定为 10 页，首尾地址也固定。

```

Heap::Heap(){
    initialize();
}

void Heap::initialize(){

    this->count=10;
    this->startAddr=0x4070000;
    this->shengyu=40960;
    this->tail=startAddr+40960;

    printf("build a heap,size is 40960B,from %x
to %x.\n",startAddr,tail);
}

```

2.设计类 HeapManager 用于对 Heap 进行 malloc 和 free。

在 memory.h 中声明。私有成员为 Heap 类型的 heap 对象。这就是我们管理的堆。

```

class HeapManager{
private:
    Heap heap;
public:
    HeapManager();
    void initialize();
    int malloc(int byte);
    void free(int byte);
};

```

在 memory.cpp 中实现。其中，malloc 函数是从上一次分配的结束位置开始连续分配，free 函数是从已分配的连续字节的最末往前释放。

```

HeapManager::HeapManager(){
    initialize();
}

void HeapManager::initialize(){
    heap.initialize();
}

```

```

}

// 实现 malloc 函数
int HeapManager::malloc(int byte){
    if(byte>heap.shengyu){
        printf("space is not enough!\n");
        return -1;
    }
    heap.startAddr+=byte;
    heap.shengyu-=byte;
    printf("malloc from %x to %x, now from %x to %x is
available.\n",heap.startAddr-
byte,heap.startAddr,heap.startAddr,heap.tail);
    return heap.startAddr-byte;
}

// 实现 free 函数
void HeapManager::free(int byte){
    if(byte>40960-heap.shengyu){
        printf("out of range!\n");
        return;
    }
    heap.startAddr-=byte;
    heap.shengyu+=byte;
    printf("free from %x to %x, now from %x to %x is
available.\n",heap.startAddr,heap.startAddr+byte,heap.startAddr,heap.ta
il);
}

```

3.测试。

在 set_up.cpp 初始化，因为我们是占用了 user pool 的前十页内存做的堆，所以我们提前分配掉前十页，避免页分配时会起冲突。

// 内存管理器

```

memoryManager.initialize();
int heap=memoryManager.allocatePhysicalPages(AddressPoolType::USER,10);
heapManager.initialize();

```

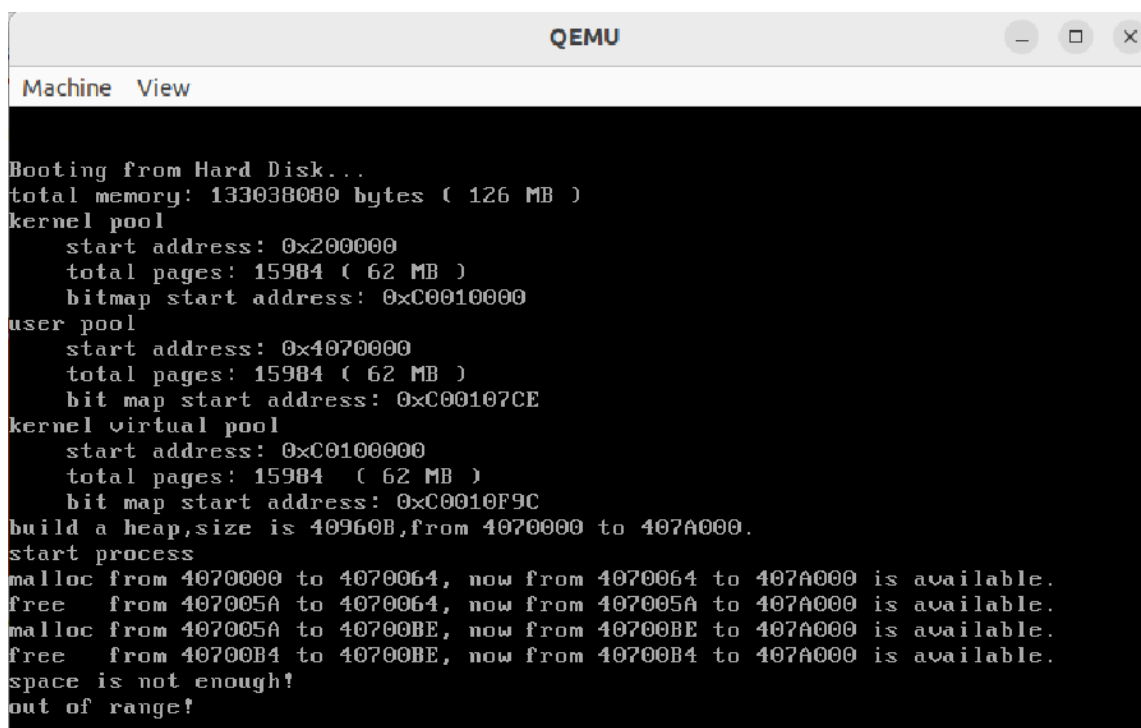
然后，在 first_process 中设计测试样例。

```

void first_process()
{
    int startAddr=heapManager.malloc(100);
    heapManager.free(10);
    startAddr=heapManager.malloc(100);
    heapManager.free(10);
    startAddr=heapManager.malloc(100000);
    heapManager.free(1000);
}

```

得到输出如下。可以看到分配和释放是严格按照先进后出原则进行的。

A screenshot of a QEMU virtual machine window. The window title is "QEMU" and it has standard window controls. Below the title bar, there is a tab labeled "Machine View". The main area of the window is a black terminal with white text. The text shows the boot process of a virtual machine, including memory pool initialization and heap management. The output indicates that memory is allocated and freed in a last-in, first-out (LIFO) manner, as seen with the "malloc" and "free" operations on the heap. The final output shows "space is not enough!" and "out of range!", indicating a memory management error.

```
Machine View

Booting from Hard Disk...
total memory: 133038080 bytes ( 126 MB )
kernel pool
  start address: 0x200000
  total pages: 15984 ( 62 MB )
  bitmap start address: 0xC0010000
user pool
  start address: 0x4070000
  total pages: 15984 ( 62 MB )
  bit map start address: 0xC00107CE
kernel virtual pool
  start address: 0xC0100000
  total pages: 15984 ( 62 MB )
  bit map start address: 0xC0010F9C
build a heap,size is 40960B,from 4070000 to 407A000.
start process
malloc from 4070000 to 4070064, now from 4070064 to 407A000 is available.
free  from 407005A to 4070064, now from 407005A to 407A000 is available.
malloc from 407005A to 40700BE, now from 40700BE to 407A000 is available.
free  from 40700B4 to 40700BE, now from 40700B4 to 407A000 is available.
space is not enough!
out of range!
```

Section 5 实验总结与心得体会

这个实验也不知道我做的符不符合要求，不过还是感慨颇多。这门课程终于结束啦！一开始完成实验还是有点吃力需要到处求助的，慢慢地我可以逐渐克服困难独立完成，还是收获了慢慢的成就感的。对操作系统的认识逐步增加的过程中，也是对我们的专业知识理解更深了，更加体会到自己是一个学计算机的学生，虽然仍然很菜喵喵喵。希望我的操作系统原理最后的分不要那么低，希望我的实验没有 A-。感谢老师和助教一个学期的认真教导，给我多多的平时分吧！